Title: Agent Based Modeling for Multi-threat Environments

Author(s): Butts, David Joshua

Intended for: Talk for modeling group

Issued: 2018-10-04

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Agent Based Modeling for Multi-threat Environments

David J Butts

Nambe
August 7th 2018

UNCLASSIFIED

Agenda

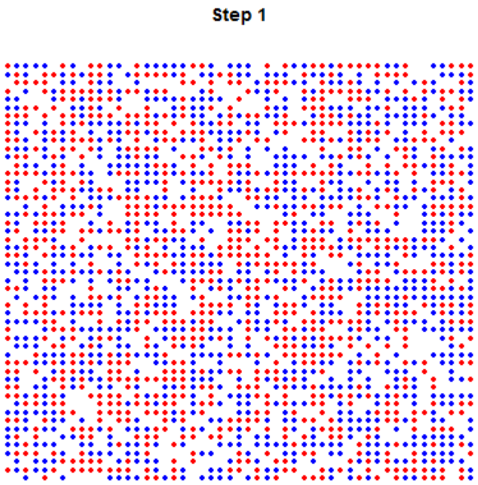
- What is agent based modeling
  - Why we use them
  - What are their pitfalls
- Examples of ABMs
  - 1D Diffusion
  - Sand pile
- My evacuation simulation project
  - Overview
  - Design
  - External Codes
  - Example output
  - Future work

Agent Based Modeling

- A simulation method that models individual ‘agents’ instead of solving equations
- Agents can be any object in a simulation
  - Person
  - Environment
  - Particle
  - Virus
- Agents follow rules to update their state
  - Move to random cell, keeping certain distance from other agents

Why do we use ABMs?

- Complexity
  - Age
  - Spatial data
- Emergent behavior
- Human systems
- Stochastic processes

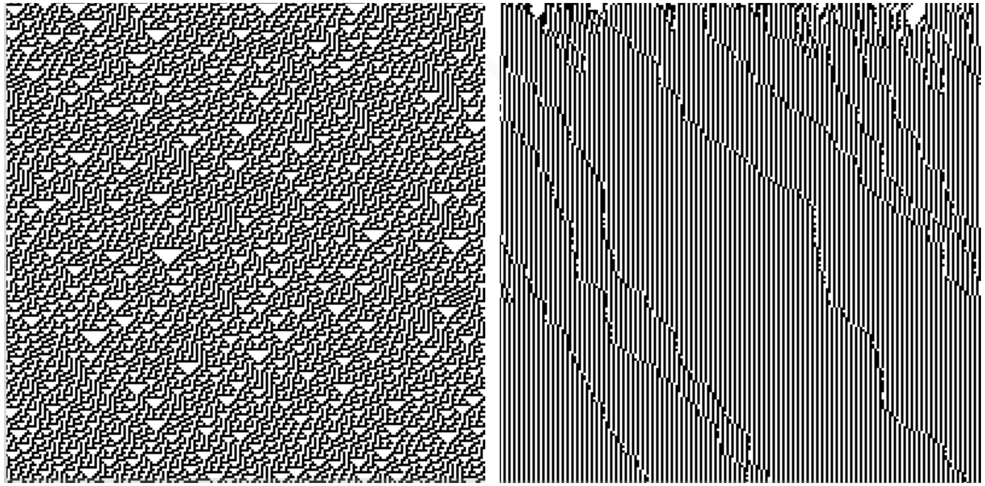


An example of a Schelling Model which shows segregation behavior emerge from a set of simple rules.

Source: http://rsnippets.blogspot.com/2012/04/animating-schellings-segregation-model.html

Issues

- Validation and Verification can be difficult
  - Finding data
  - Behaviors can look right, does not mean they are
- Very sensitive to ‘small’ choices
  - Updating sequentially vs randomly
  - Choice of distributions to sample
  - Dimensionality

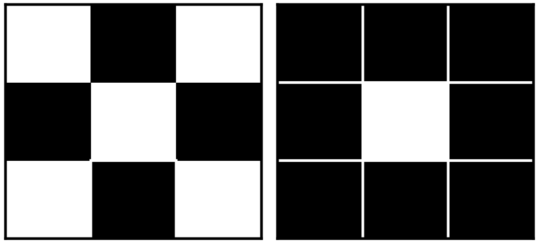


The same code can give very different results based on which updating rule is used. On the left, each cell is update simultaneously. On the right, each cell is updated in a random order. This simple change in updating completely changes the behavior of the solution.

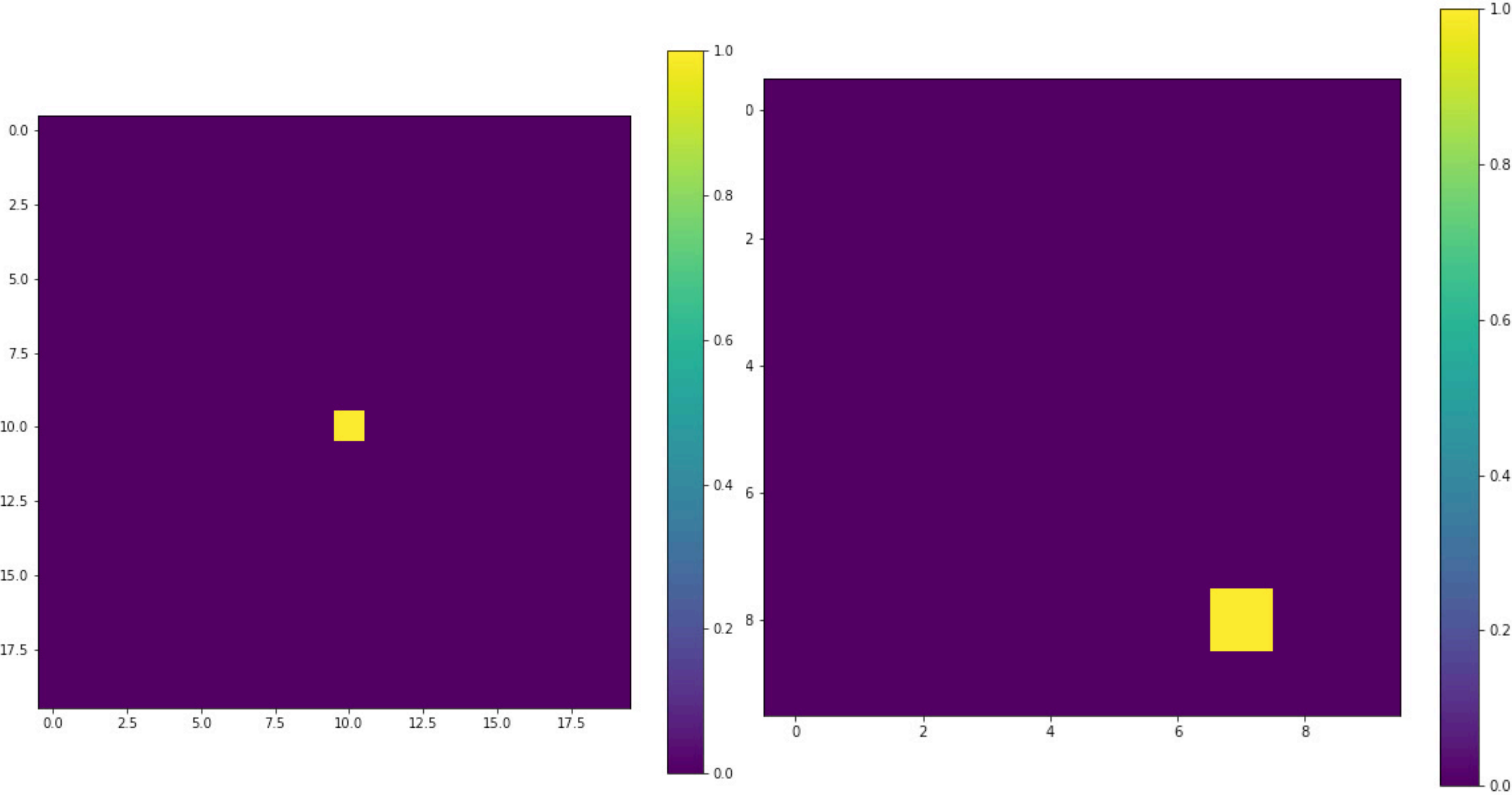
Source: https://en.wikipedia.org/wiki/Asynchronous_cellular_automaton

Sand Pile Model

- Rules
  - Randomly place grains of sand on a board one at a time
  - If 4 land in the same place, spill into the Von Neumann neighborhood
  - Sand falls on the edge of the board
- Useful for studying catastrophic event
  - Display self-organized criticality
  - Fat tail statistics



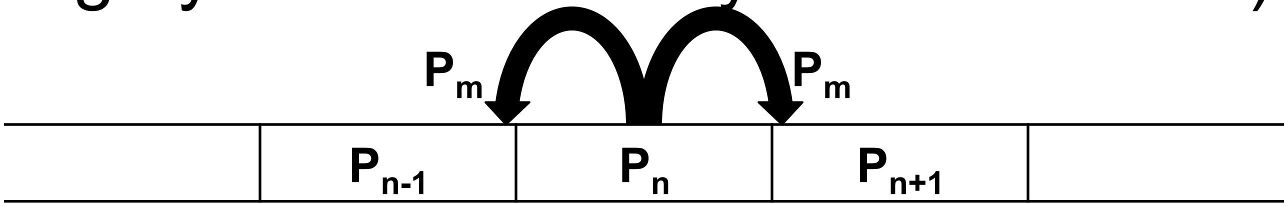
A comparison of Von Neumann (left) and Moore (right) neighborhoods.



On the right the sand is always dropped in the center of the board, while on the left it is dropped randomly. You can see as more sand piles up, larger regions are effected when a sand pile collapses. It is possible for one grain of sand to cause the entire board to be effected.

Model Equivalence: 1D Diffusion

- If care is taken you can create agent based models that model the real world and should have equivalent mathematical models. (though you cannot always write it down!)



- A particle at P_n will randomly move left or right with equal probability

Relating ABM Rules to Physical Behavior

$$P_n(t+1) = P_n(t) + P_m P_{n-1}(t) + P_m P_{n+1}(t) - 2P_m P_n(t)$$

$$P(x,t+1) - P(x,t) = P_m P(x-1,t) + P_m P(x+1,t) - 2P_m P(x,t)$$

$$\frac{\partial P}{\partial t} + \ldots = P_m \frac{\partial^2 P}{\partial^2 x} + \ldots$$

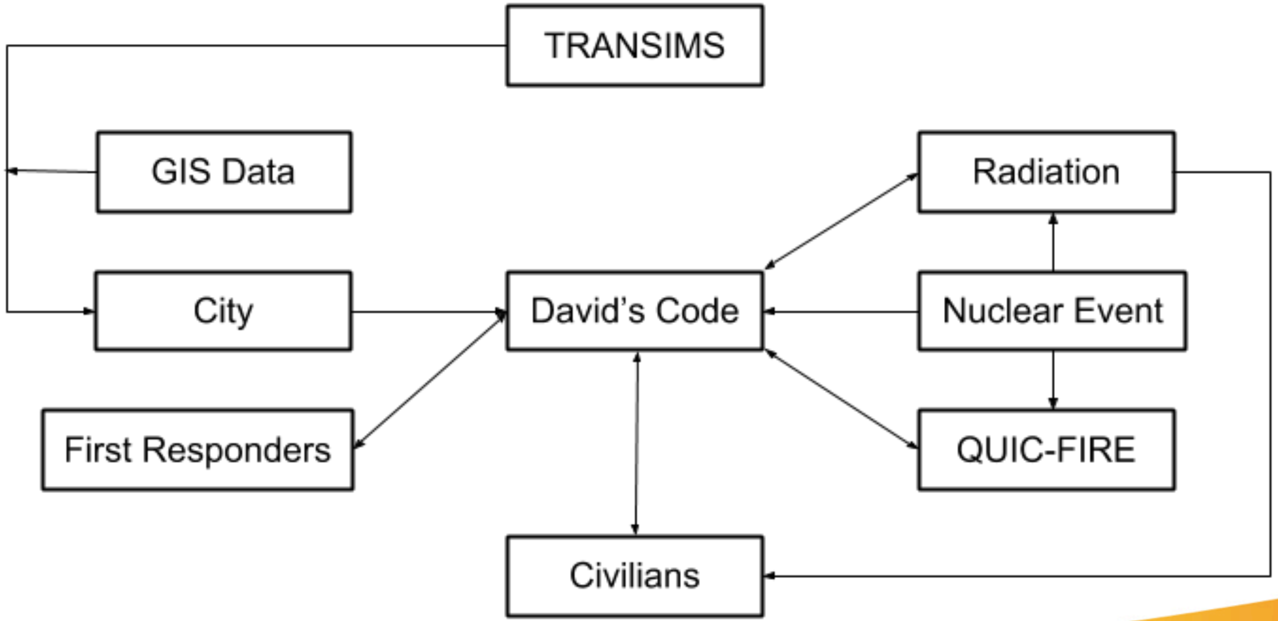
- Starting with the probability of a particle being in P_n at the next time step, and Taylor expand keeping to the first surviving terms, we arrive at the diffusion equation.

Creating an Evacuation Simulation Model

- We are developing a tool to simulate multi-threat environments to use in evacuation efforts during extreme events
- Many codes simulate the individual pieces of a disaster
  - Fire
  - Damage to buildings
  - Radiation
- We are bringing together the many codes and micro simulating human behavior in these situations to optimize
  - Evacuation plans
  - Rescue plans
  - First responders response

Designing the Code Structure

- The code consists of multiple modules
- Each module can be replaced by custom ones
- Data stored as binary files



Linking the Codes

- Module Requirements
  - The code needs to be wrapped with Python
  - Must return path to array it edits
- Three options for modules
  - Can be a Python code that directly edits files
  - Can call an external code to edit files
  - Can return the next file in a list of recalculated ones

Module I am using

- I plan to use a precalculated data for the radiation layer
- Working on incorporating TRANSIMS for transportation
- Incorporated QUIC-Fire for fire layer

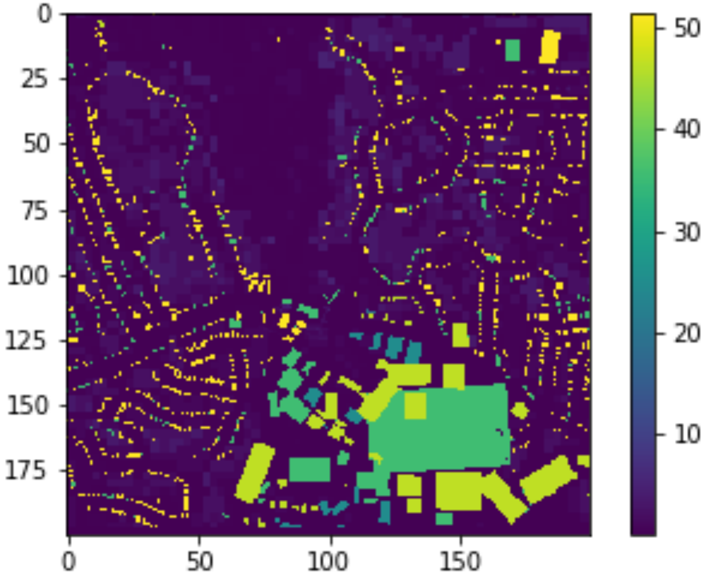
TRANSIMS

- TRansportation ANalysis SIMulation System
- Cellular Automata based traffic micro-simulator
- Takes in various census and GIS data
- Creates routes

https://www.youtube.com/watch?v=uL5mFuQv-bc#action=share

QUIC-Fire

- CA fire simulator developed at LANL
- Has many input files
  - QUIC_fire.inp
  - QU_simparams.inp
  - QF_FuelDensity.inp
  - QF_IgnitionPattern.inp
  - QF_FuelMoisture.inp



An example of what the data in QF_FuelDensity.inp looks like. QF_FuelMoisture has similar data.

Issues with QUIC-Fire

- No methods for pausing calculations
  - Cannot interact with fire after a simulation has started
- A new fire can be initialized using data from a previous run
- This allows for a dynamic simulation, but introduces error
  - Well established fire restarts as a newly starting fire

Agents

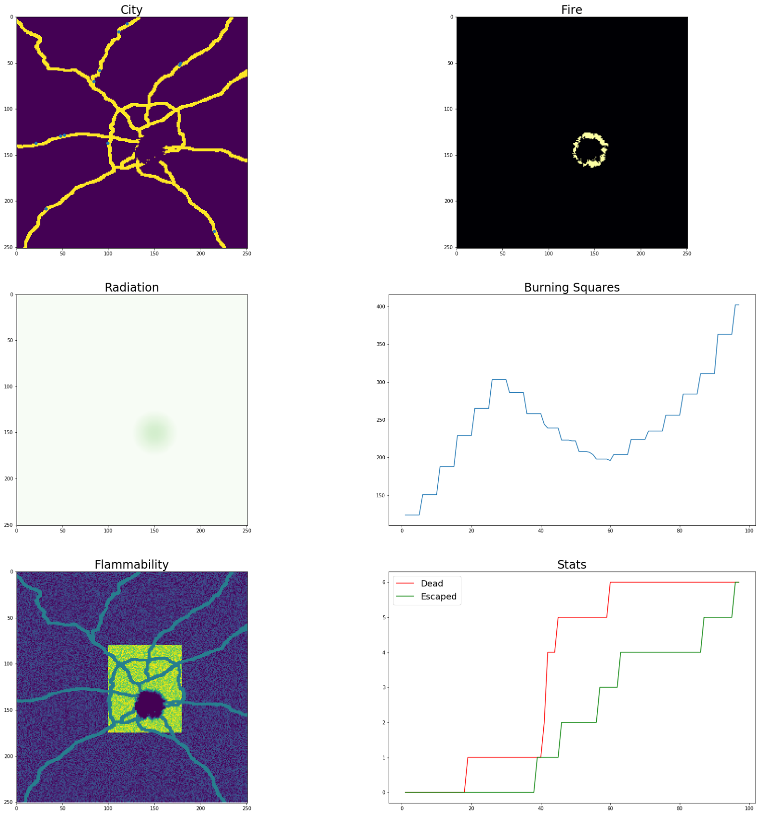
- Agent requirements
  - Python objects named Car, Civilian, or Firefighter
  - Four methods
    - __init__(x,y,strategy)
      - Need a boolean alive attribute
    - move()
    - interact()
      - Agent death happens here
    - draw()

My Agents

- Civilian
  - Wait, random motion, run radially
  - Die from fire and radiation
- Firefighter
  - Random motion, aggressive, avoidance
  - Die from fire and radiation
- Car
  - Probabilistic movement on roads
  - Die from fire and radiation

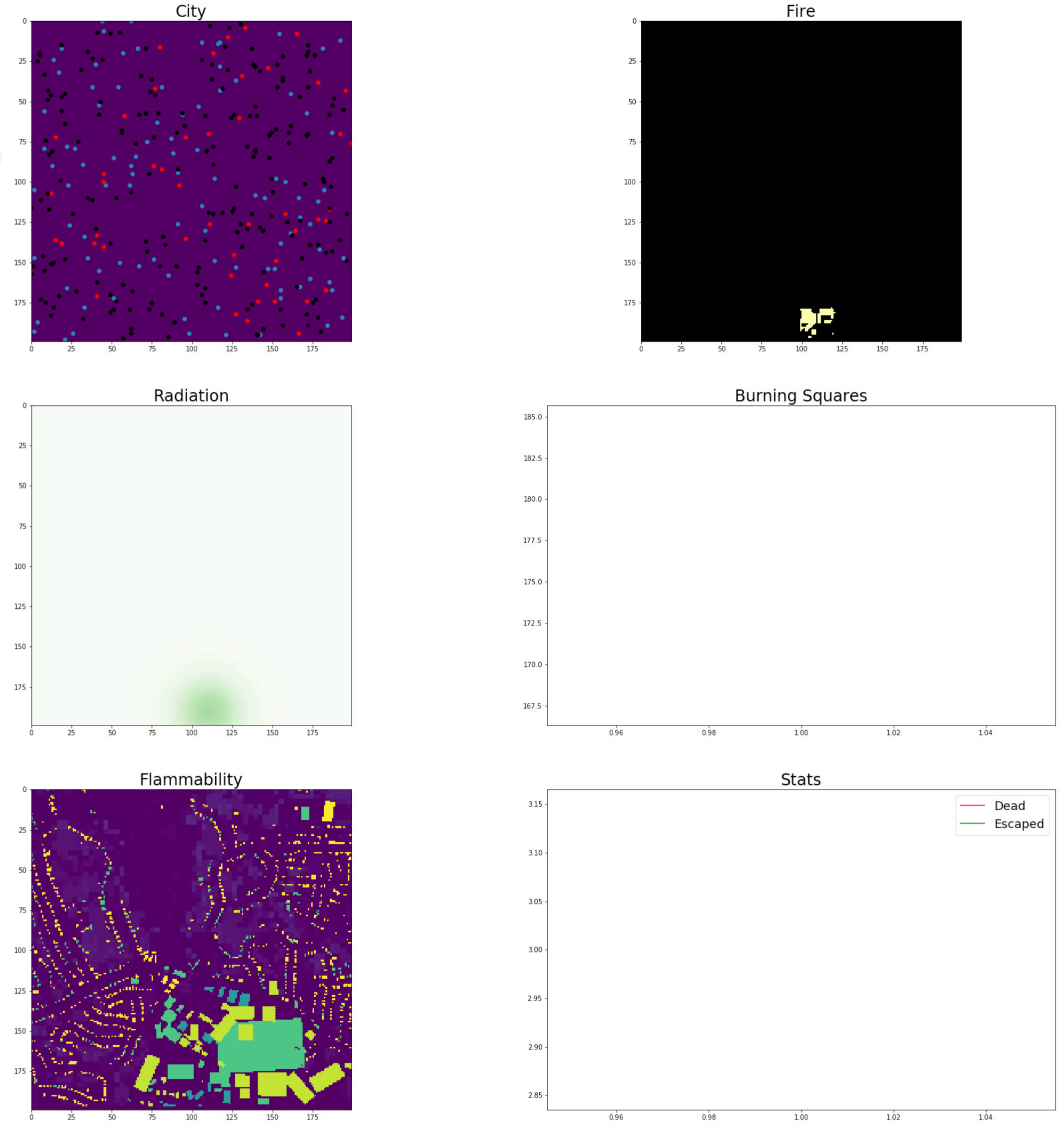
Code Outputs

- Each layer is saved as an array at its final state
- The code tracks
  - Burning cells
  - Escaped agents
  - Dead agents
- Single layer or grid
- Saves output each time step for making movies



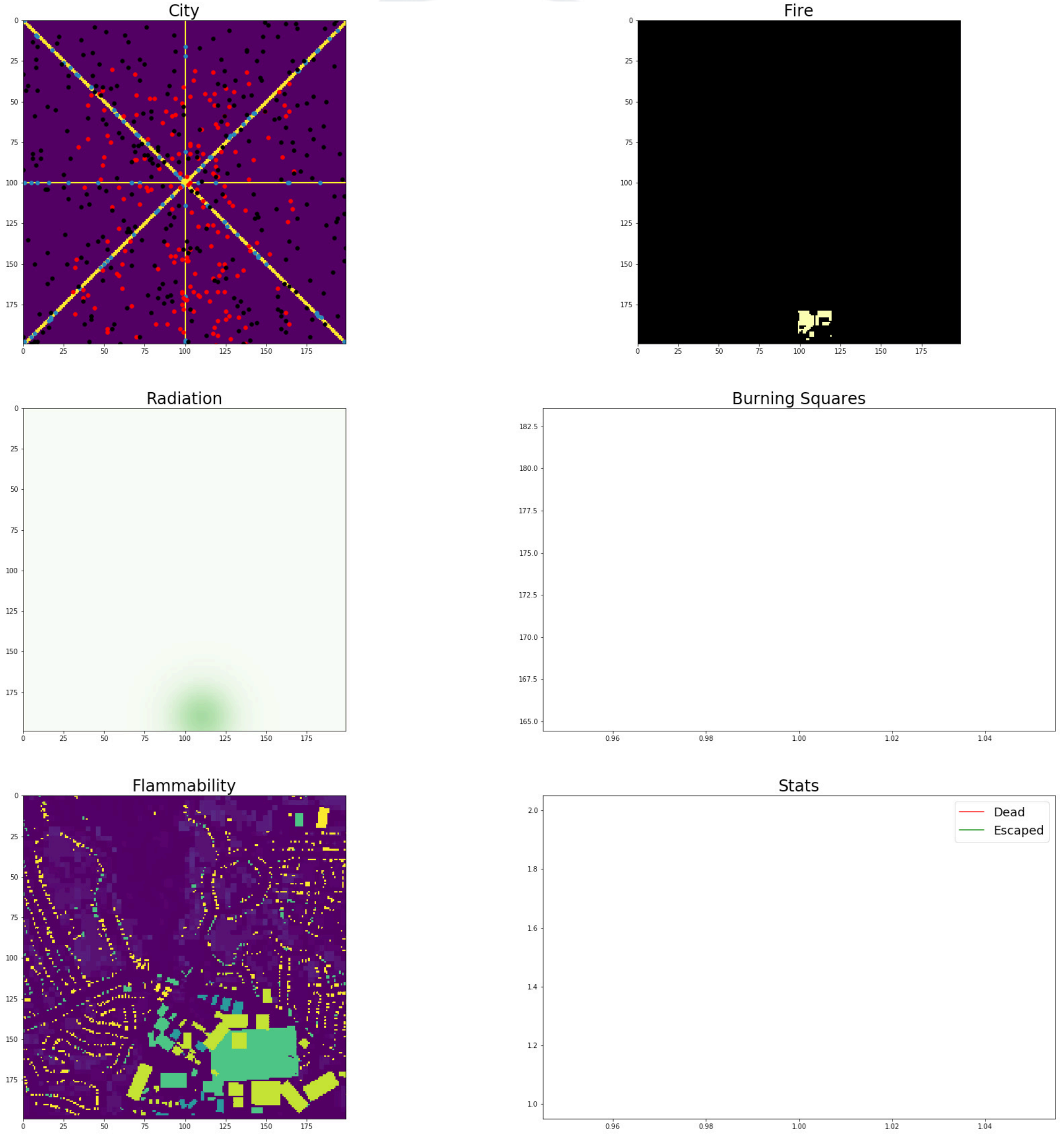
words

Random Motion Example



UNCLASSIFIED

Basic Strategy Example



City

Fire

Radiation

Burning Squares

Flammability

Stats

UNCLASSIFIED

Future Work

- More realistic movement for agents
- More outputs
  - Evacuation times
  - Percent burned
- Incorporating TRANSIMS and real radiation data
- Parallelization with mpi4py

Acknowledgments

- Jim Cooley for
  - Supporting me as a summer student
  - Getting me set up at the lab
  - Creating contacts to other collaborators
  - Helping me troubleshoot my codes
- Michael Murillo for
  - Teaching me about agent based modeling
  - Helping me troubleshoot my codes
- Michael Brown for
  - providing me with the QUIC code and helping me understand how to use it
- Eunmo Koo for
  - Providing data sets for fire simulations